# Green Bank Observatory

# LASSI
Software Architecture Document (SAD)

**CONTENT OWNER: Nathaniel D. Sizemore (nsizemor@nrao.edu)**

| Version | Date | Notes/changes |
|---|---|---|
| 0.01 | | Initial draft to stimulate discussion |

**BACKGROUND**

This template is based on the Software Engineering Institute's "View and Beyond" method for documenting software architectures, as described in Clements, et al., *Documenting Software Architecture: Views and Beyond* (Addison Wesley, 2002). The current version is available for free download from the SEI's architecture web site.

**TIPS FOR USING THIS TEMPLATE**

To create an instance of this document:

- Insert relevant information on cover sheet and in placeholders throughout.
- Insert relevant information in page header: Move to a page of the body of the report, select *View > Header and Footer* from the main menu, and then replace relevant information in the header box at the top of the page.

To update the contents and page numbers in the Table of Contents, List of Figures, and List of Tables:

- Position the cursor anywhere in the table to be updated.
- Click the *F9* function key.
- Answer "Update entire table".

To insert a figure or table caption:

- From the main menu, choose *Insert > Reference > Caption* and then either *Figure* or *Table* as needed.
- Click the OK button.
- Add a colon and a tab stop after the figure number in the caption itself.
- The caption should use the *Caption* style.
- Add a colon and a tab stop after the table/figure number in the caption itself.

**TIPS FOR MAKING YOUR DOCUMENT MORE READABLE**

- A gray box containing *CONTENTS OF THIS SECTION* is provided at the beginning of most sections and subsections. After determining what specific information will be included in your document, you can remove this gray box or leave it to serve as a quick-reference section overview for your readers. In the case that text has been provided in the template, inspect it for relevance and revised as necessary.
- Consider hyperlinking key words used in the document with their entries in the Glossary or other location in which they are defined. Choose *Insert > Hyperlink*.
- Don't leave blank sections in the document. Mark them "To be determined" (ideally with a promise of a date or release number by which the information will be provided) or "Not applicable."
- Consider packaging your SAD as a multi-volume set of documentation. It is often helpful to break your documentation into more than one volume so that the document does not become unwieldy. There are many ways that this can be accomplished. The structuring of the document must support the needs of the intended audience and must be determined in the context of the project. Each document that you produce should include the date of issue and status; draft, baseline, version number, name of issuing organization; change history; and a summary. A few decomposition options are:
  - *A 2-Volume approach:* Separate the documentation into two volumes; one that contains the views of the software architecture and one that contains everything else. A common variant of this approach has one volume per view, and one volume for everything else.
  - *A 3-Volume approach:* Document organizational policies, procedures, and the directory in one volume, system specific overview material in a second, and view documentation in a third.
  - *A 4-Volume approach:* Create one volume for each viewtype [Clements 2002] (module, component-and-connector, allocation) that contains the documentation for the relevant views. Include all of the other information in the fourth volume.
  - Software interfaces are often documented in a separate volume.

  In *any* case, the information should be arranged so that readers begin with the volume containing the Documentation Roadmap (Section 1 in this template).

## Table of Contents

# List of Figures

# List of Tables

# 1 Documentation Roadmap

This document describes the LASSI software architecture, and is organized as follows:

- Section 1.1 (“Document Management and Configuration Control Information”) explains revision history. This tells you if you’re looking at the correct version of the SAD.
- Section 1.2 (“Purpose and Scope of the SAD”) explains the purpose and scope of the SAD, and indicates what information is and is not included. This tells you if the information you’re seeking is likely to be in this document.
- Section 1.3 (“How the SAD Is Organized”) explains the information that is found in each section of the SAD. This tells you what section(s) in this SAD are most likely to contain the information you seek.
- Section 1.4 (“Stakeholder Representation”) explains the stakeholders for which the SAD has been particularly aimed. This tells you how you might use the SAD to do your job.
- Section 1.5 (“How a View is Documented”) explains the standard organization used to document architectural views in this SAD. This tells you what section within a view you should read in order to find the information you seek.

## 1.1 Document Management and Configuration Control Information

- Revision Number: 0.1
- Revision Release Date: TBD
- Purpose of Revision: Initial draft to stimulate discussion
- Scope of Revision: *new document from SEI template.*

## 1.2 Purpose and Scope of the SAD

**CONTENTS OF THIS SECTION**: This section explains the SAD’s overall purpose and scope, the criteria for deciding which design decisions are architectural (and therefore documented in the SAD), and which design decisions are non-architectural (and therefore documented elsewhere).

This SAD specifies the software architecture for the LASSI project**.** All information regarding the software architecture may be found in this document, although much information is incorporated by reference to other documents.

**What is software architecture?** The software architecture for a system[1] is the structure or structures of that system, which comprise software elements, the externally-visible properties of those elements, and the relationships among them [Bass 2003]. "Externally visible" properties refers to those assumptions other elements can make of an element, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on. This definition provides the basic litmus test for what information is included in this SAD, and what information is relegated to downstream documentation.

**Elements and relationships**. The software architecture first and foremost embodies information about how the elements relate to each other. This means that architecture specifically omits certain information about elements that does not pertain to their interaction. Thus, a software architecture is an *abstraction* of a system that suppresses details of elements that do not affect how they use, are used by, relate to, or interact with other elements. Elements interact with each other by means of interfaces that partition details about an element into public and private parts. Software architecture is concerned with the public side of this division, and that will be documented in this SAD accordingly. On the other hand, private details of elements—details having to do solely with internal implementation—are not architectural and will not be documented in a SAD.

**Multiple structures.** The definition of software architecture makes it clear that systems can and do comprise more than one structure and that no one structure holds the irrefutable claim to being the architecture. The neurologist, the orthopedist, the hematologist, and the dermatologist all take a different perspective on the structure of a human body. Ophthalmologists, cardiologists, and podiatrists concentrate on subsystems. And the kinesiologist and psychiatrist are concerned with different aspects of the entire arrangement's behavior. Although these perspectives are pictured differently and have very different properties, all are inherently related; together they describe the architecture of the human body. So it is with software. Modern systems are more than complex enough to make it difficult to grasp them all at once. Instead, we restrict our attention at any one moment to one (or a small number) of the software system's structures. To communicate meaningfully about an architecture, we must make clear which structure or structures we are discussing at the moment—which *view* we are taking of the architecture. Thus, this SAD follows the principle that documenting a software architecture is a matter of documenting the relevant views and then documenting information that applies to more than one view.

For example, all non-trivial software systems are partitioned into implementation units; these units are given specific responsibilities, and are the basis of work assignments for programming teams. This kind of element will comprise programs and data that software in other implementation units can call or access, and programs and data that are private. In large projects, the elements will almost certainly be subdivided for assignment to sub-teams. This is one kind of structure often used to describe a system. It is a very static structure, in that it focuses on the way the system's functionality is divided up and assigned to implementation teams.

[1] Here, a system may refer to a system of systems.

Other structures are much more focused on the way the elements interact with each other at runtime to carry out the system's function. Suppose the system is to be built as a set of parallel processes. The set of processes that will exist at runtime, the programs in the various implementation units described previously that are strung together sequentially to form each process, and the synchronization relations among the processes form another kind of structure often used to describe a system.

None of these structures alone is *the* architecture, although they all convey architectural information. The architecture consists of these structures as well as many others. This example shows that since architecture can comprise more than one kind of structure, there is more than one kind of element (e.g., implementation unit and processes), more than one kind of interaction among elements (e.g., subdivision and synchronization), and even more than one context (e.g., development time versus runtime). By intention, the definition does not specify what the architectural elements and relationships are. Is a software element an object? A process? A library? A database? A commercial product? It can be any of these things and more.

These structures will be represented in the views of the software architecture that are provided in Section 3.

**Behavior.** Although software architecture tends to focus on structural information, *behavior of each element is part of the software architecture* insofar as that behavior can be observed or discerned from the point of view of another element. This behavior is what allows elements to interact with each other, which is clearly part of the software architecture and will be documented in the SAD as such. Behavior is documented in the element catalog of each view.

## 1.3 How the SAD Is Organized

**CONTENTS OF THIS SECTION**: This section provides a narrative description of the major sections of the SAD and the overall contents of each. Readers seeking specific information can use this section to help them locate it more quickly.

This SAD is organized into the following sections:

- **Section 1 ("Documentation Roadmap") provides information about this document and its intended audience**. It provides the roadmap and document overview. Every reader who wishes to find information relevant to the software architecture described in this document should begin by reading Section 1, which describes how the document is organized, which stakeholder viewpoints are represented, how stakeholders are expected to use it, and where information may be found. Section 1 also provides information about the views that are used by this SAD to communicate the software architecture.
- **Section 2 ("Architecture Background") explains why the architecture is what it is.** It provides a system overview, establishing the context and goals for the development. It describes the background and rationale for the software architecture. It explains the constraints and influences that led to the current architecture, and it describes the major architectural approaches that have been utilized in the architecture. It includes information

about evaluation or validation performed on the architecture to provide assurance it meets its goals.

- **Section 3 (Views") and Section 4 ("Relations Among Views") specify the software architecture**. Views specify elements of software and the relationships between them. A view corresponds to a viewpoint (see Section Error: Reference source not found), and is a representation of one or more structures present in the software (see Section 1.2).
- **Sections 5 ("Referenced Materials") and 6 ("Directory") provide reference information for the reader.** Section 5 provides look-up information for documents that are cited elsewhere in this SAD. Section 6 is a *directory*, which is an index of architectural elements and relations telling where each one is defined and used in this SAD. The section also includes a glossary and acronym list.

## 1.4 Stakeholder Representation

This section provides a list of the stakeholder roles considered in the development of the architecture described by this SAD. For each, the section lists the concerns that the stakeholder has that can be addressed by the information in this SAD.

Each stakeholder of a software system—customer, user, project manager, coder, analyst, tester, and so on—is concerned with different characteristics of the system that are affected by its software architecture. For example, the user is concerned that the system is reliable and available when needed; the customer is concerned that the architecture can be implemented on schedule and to budget; the manager is worried (in addition to cost and schedule) that the architecture will allow teams to work largely independently, interacting in disciplined and controlled ways. The developer is worried about strategies to achieve all of those goals. The security analyst is concerned that the system will meet its information assurance requirements, and the performance analyst is similarly concerned with it satisfying real-time deadlines.

This information is represented as a matrix, where the rows list stakeholder roles, the columns list concerns, and a cell in the matrix contains an indication of how serious the concern is to a stakeholder in that role. This information is used to motivate the choice of viewpoints chosen in Section Error: Reference source not found.

| | | |
|---|---|---|
| • | • | • |

## 1.5 How a View is Documented

Section 3 of this SAD contains one view for each viewpoint listed in Section 1.5. Each view is documented as a set of view packets. A view packet is the smallest bundle of architectural documentation that might be given to an individual stakeholder.

Each view is documented as follows, where the letter *i* stands for the number of the view: 1, 2, etc.:

- Section 3.i: Name of view.
- Section 3.i.1: View description. This section describes the purpose and contents of the view. It should refer to (and match) the viewpoint description in Section 1.5 to which this view conforms.
- Section 3.i.4: Variability mechanisms. This section describes any architectural variability mechanisms (e.g., adaptation data, compile-time parameters, variable replication, and so forth) described by this view, including a description of how and when those mechanisms may be exercised and any constraints on their use.
- Section 3.i.5.j.1: Primary presentation. This section presents the elements and the relations among them that populate this view packet, using an appropriate language, languages, notation, or tool-based representation.
  - Section 3.i.5.j.2: Element catalog. Whereas the primary presentation shows the important elements and relations of the view packet, this section provides additional information needed to complete the architectural picture. It consists of the following subsections:
  - Section 3.i.5.j.2.1: Elements. This section describes each element shown in the primary presentation, details its responsibilities of each element, and specifies values of the elements' relevant *properties*, which are defined in the viewpoint to which this view conforms.
  - Section 3.i.5.j.2.2: Relations. This section describes any additional relations among elements shown in the primary presentation, or specializations or restrictions on the relations shown in the primary presentation.
  - Section 3.i.5.j.2.3: Interfaces. This section specifies the software interfaces to any elements shown in the primary presentation that must be visible to other elements.
  - Section 3.i.5.j.2.4: Behavior. This section specifies any significant behavior of elements or groups of interacting elements shown in the primary presentation.
  - Section 3.i.5.j.2.5: Constraints: This section lists any constraints on elements or relations not otherwise described.
  - Section 3.i.5.j.3: Context diagram. This section provides a context diagram showing the context of the part of the system represented by this view packet. It also designates the view packet's scope with a distinguished symbol, and shows interactions with external entities in the vocabulary of the view.
  - Section 3.i.5.j.4: Variability mechanisms. This section describes any variabilities that are available in the portion of the system shown in the view packet, along with how and when those mechanisms may be exercised.
  - Section 3.i.5.j.5: Architecture background. This section provides rationale for any significant design decisions whose scope is limited to this view packet.
  - Section 3.i.5.j.6: Relation to other view packets. This section provides references for related view packets, including the parent, children, and siblings of this view packet. Related view packets may be in the same view or in different views.

## 1.6 Process for Updating this SAD

If you find errors, ambiguities, or other content that needs to be updated (particularly sections marked “TBD”), please contact the document author, Nathaniel D. Sizemore, at nsizemor@nrao.edu. The most recent published version of this document will be available on the LASSI SharePoint site.

# 2 Architecture Background

## 2.1 Problem Background

**CONTENTS OF THIS SECTION**: The sub-parts of Section 2.1 explain the constraints that provided the significant influence over the architecture.

The LASSI architecture is designed to fulfill the goals stated in the GBT Metrology ConOps document, available on SharePoint.

### 2.1.1 System Overview

**CONTENTS OF THIS SECTION**: This section describes the general function and purpose for the system or subsystem whose architecture is described in this SAD.

This architecture describes the system for controlling the TLS device, acquiring data, performing post-processing, calculating needed values, and giving those values to the GBT M&C system.

### 2.1.2 Goals and Context

**CONTENTS OF THIS SECTION**: This section describes the goals and major contextual factors for the software architecture. The section includes a description of the role software architecture plays in the life cycle, the relationship to system engineering results and artifacts, and any other relevant factors.

For the complete context and goals, see the ConOps document, referenced in section 2.1.

### 2.1.3 Significant Driving Requirements

**CONTENTS OF THIS SECTION**: This section describes behavioral and quality attribute requirements (original or derived) that shaped the software architecture. Included are any scenarios that express driving behavioral and quality attribute goals, such as those crafted during a Quality Attribute Workshop (QAW) [Barbacci 2003] or software architecture evaluation using the Architecture Tradeoff Analysis Method$^{SM}$ (ATAM$^{SM}$) [Bass 2003].

Please review the ConOps document referenced in section 2.1.1 for driving requirements.

## 2.2 Solution Background

**CONTENTS OF THIS SECTION**: The sub-parts of Section 2.2 provide a description of why the architecture is the way that it is, and a convincing argument that the architecture is the right one to satisfy the behavioral and quality attribute goals levied upon it.

TBD.

S M Quality Attribute Workshop and QAW and Architecture Tradeoff Analysis Method and ATAM are service marks of Carnegie Mellon University.

### 2.2.1 Architectural Approaches

**CONTENTS OF THIS SECTION**: This section provides a rationale for the major design decisions embodied by the software architecture. It describes any design approaches applied to the software architecture, including the use of architectural styles or design patterns, when the scope of those approaches transcends any single architectural view. The section also provides a rationale for the selection of those approaches. It also describes any significant alternatives that were seriously considered and why they were ultimately rejected. The section describes any relevant COTS issues, including any associated trade studies.

LASSI requires that data flow from the scanner hardware, through a series of manipulations and reductions, ending in commands to the GBT's active surface. This workflow maps to the common pipe-and-filter architecture pattern.

### 2.2.2 Analysis Results

**CONTENTS OF THIS SECTION**: This section describes the results of any quantitative or qualitative analyses that have been performed that provide evidence that the software architecture is fit for purpose. If an Architecture Tradeoff Analysis Method evaluation has been performed, it is included in the analysis sections of its final report. This section refers to the results of any other relevant trade studies, quantitative modeling, or other analysis results.

TBD – perhaps note our prototyping here, either current work or Fred's past studies.

### 2.2.3 Requirements Coverage

**CONTENTS OF THIS SECTION**: This section describes the requirements (original or derived) addressed by the software architecture, with a short statement about where in the architecture each requirement is addressed.

Refer to the GBT Metrology Architecture Plan for details on the current requirements.

### 2.2.4 Summary of Background Changes Reflected in Current Version

**CONTENTS OF THIS SECTION**: For versions of the SAD after the original release, this section summarizes the actions, decisions, decision drivers, analysis and trade study results that became decision drivers, requirements changes that became decision drivers, and how these decisions have caused the architecture to evolve or change.

TBD

# 3 Views

This section contains the views of the software architecture. A view is a representation of a whole system from the perspective of a related set of concerns [IEEE 1471]. Concretely, a view shows a particular type of software architectural elements that occur in a system, their properties, and the relations among them. A view conforms to a defining viewpoint.

Architectural views can be divided into three groups, depending on the broad nature of the elements they show. These are:

- Module views. Here, the elements are modules, which are units of implementation. Modules represent a code-based way of considering the system. Modules are assigned areas of functional responsibility, and are assigned to teams for implementation. There is less emphasis on how the resulting software manifests itself at runtime. Module structures allow us to answer questions such as: What is the primary functional responsibility assigned to each module? What other software elements is a module allowed to use? What other software does it actually use? What modules are related to other modules by generalization or specialization (i.e., inheritance) relationships?

- Component-and-connector views. Here, the elements are runtime components (which are principal units of computation) and connectors (which are the communication vehicles among components). Component and connector structures help answer questions such as: What are the major executing components and how do they interact? What are the major shared data stores? Which parts of the system are replicated? How does data progress through the system? What parts of the system can run in parallel? How can the system's structure change as it executes?

- Allocation views. These views show the relationship between the software elements and elements in one or more external environments in which the software is created and executed. Allocation structures answer questions such as: What processor does each software element execute on? In what files is each element stored during development, testing, and system building? What is the assignment of the software element to development teams?

These three kinds of structures correspond to the three broad kinds of decisions that architectural design involves:

- How is the system to be structured as a set of code units (modules)

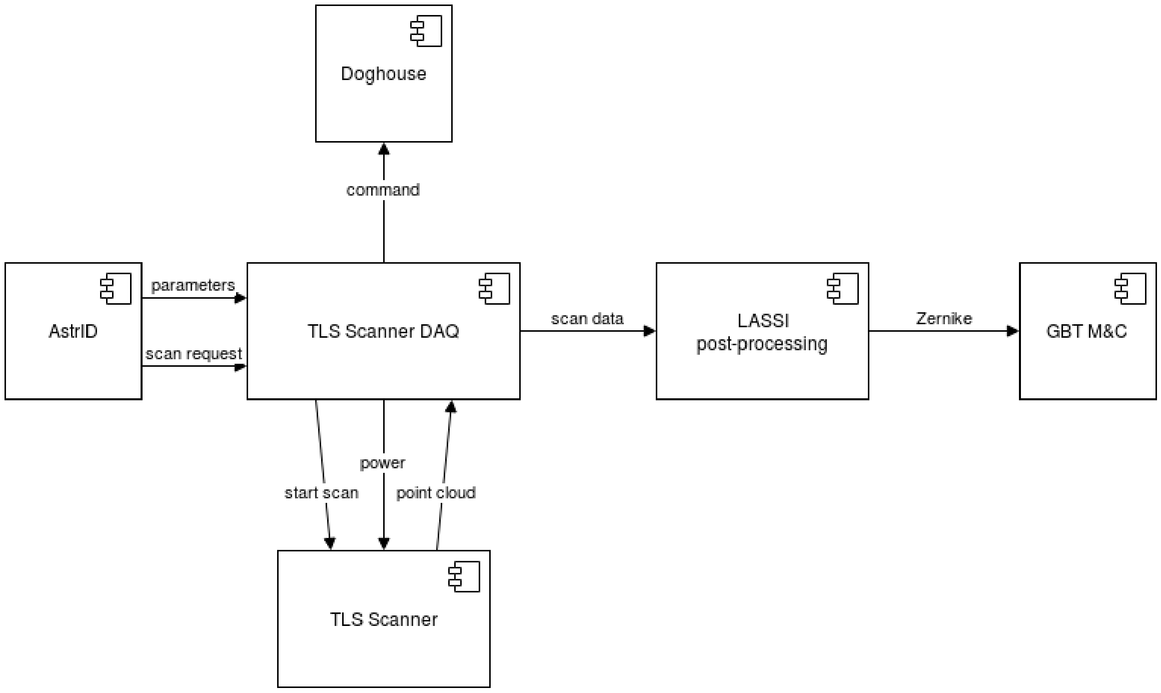
- How is the system to be structured as a set of elements that have run-time behavior (components) and interactions (connectors) ?

- How is the system to relate to non-software structures in its environment (such as CPUs, file systems, networks, development teams, etc.)?

Often, a view shows information from more than one of these categories. However, unless chosen carefully, the information in such a hybrid view can be confusing and not well understood.

## 3.1 Component-and-Connector View

### 3.1.1 Primary Presentation



### 3.1.2 Element Catalog

- AstrID – Astronomer’s Integrated Desktop
- TLS Scanner – Leica scanner hardware
- Doghouse – environmental housing for the TLS Scanner, and any other associated mechanical components required for operation on the GBT structure
- TLS Scanner DAQ – software commanding the TLS scanner and transferring data to and from it
- LASSI post-processing – GBO-developed software that takes in point clouds from the scanner, does any required data reduction, and returns calculated Zernike coefficients that describe the GBT active surface
- GBT M&C – GBT Monitor and Control system

### 3.1.3 Context Diagram

The context for LASSI in this view is encapsulated by the view itself.

### 3.1.4 Variability Mechanisms

None.

### 3.1.5 Architecture Background

This structure is based on the minimum viable product (MVP) described in the ConOps documentation, Appendix A. Note that as such, it does *not* include features such as storing the surface data as shown in Figure 3 of the ConOps.

### 3.1.6 Related Views

TBD.

## 3.2 TLS Scanner DAQ Decomposition View

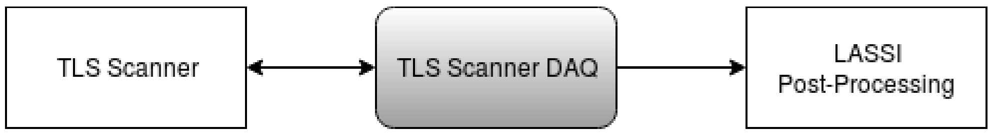
### 3.2.1 Primary Presentation

TBD

### 3.2.2 Element Catalog

- TBD

### 3.2.3 Context Diagram



### 3.2.4 Variability Mechanisms

TBD

### 3.2.5 Architecture Background

TBD

### 3.2.6 Related Views

TBD

## 3.3 LASSI Post-Processing Decomposition View

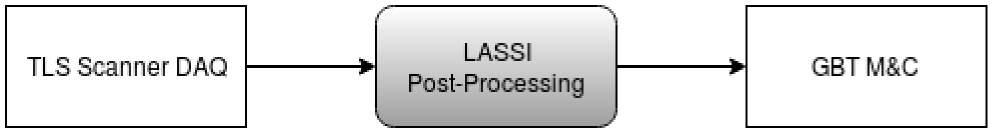
### 3.3.1 Primary Presentation

TBD

### 3.3.2 Element Catalog

- TBD

### 3.3.3 Context Diagram



### 3.3.4 Variability Mechanisms

TBD

### 3.3.5 Architecture Background

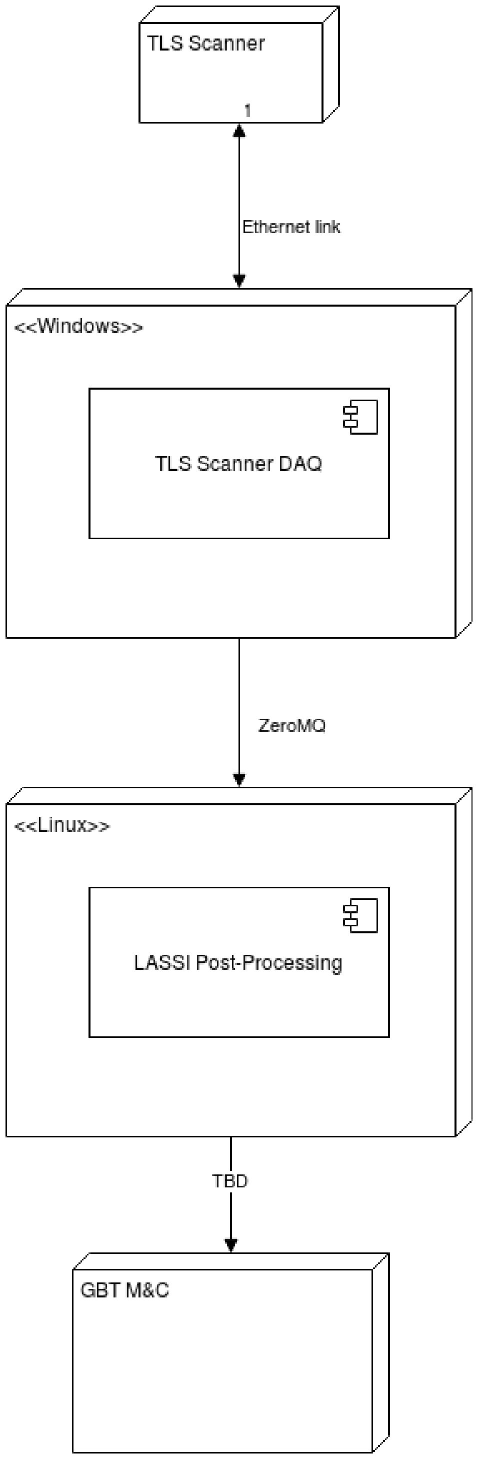
The LASSI Post-Processing component is based on Fred Schwab's original prototyping, done with Mathematica. The TLS Scanner produces a point cloud, which then must be filtered to include only relevant parts of the scan (i.e. removing any data from the TLS Scanner that is not the surface of the primary reflector of the GBT). The LASSI Post-Processing must then fit a curve to the data and calculate a series of Zernike polynomials from that curve. These Zernike polynomials are then used to both measure the difference in subsequent scans and passed to the GBT M&C system, so that the active surface can make any needed corrections.

### 3.3.6 Related Views

TBD

## 3.4 Deployment View

### 3.4.1 Primary Presentation



### 3.4.2 Element Catalog

#### Elements

- TLS Scanner – Leica hardware scanner
- Windows – computer running Windows 10; this is needed because the TLS Scanner only provides a .NET interface. This machine may also include GPU hardware, depending on the needs of the LASSI Post-Processing component.
- Linux – computer running RHEL7
- GBT M&C – monitor and control system for GBT.

#### Relationships

- Ethernet link – physical Ethernet connection using Cat6e or optical fiber with media converters
- ZeroMQ – messaging protocol allowing for multiple subscribers.

### 3.4.3 Context Diagram

TBD

### 3.4.4 Variability Mechanisms

None.

### 3.4.5 Architecture Background

Where ever possible, LASSI should be deployed on hardware running Red Hat Enterprise Linux; this is what the majority of the GBT M&C system runs on, and thus makes LASSI easier to maintain, modify, and integrate. The notable exception to this is the LASSI DAQ. The API provided by Leica for the TLS scanner is based on .NET, and must be run on Windows.

We explored ways of using the Leica API on Linux; unfortunately, some .DLLs provided by the scanner manufacturer caused illegal instructions when run under Mono. There may also be warranty or support issues from the manufacturer if we are not running on on officially supported platform. Given these issues, we have designed LASSI to be deployed with the DAQ running on a Windows host.

LASSI must be able to go from a commanded measurement to issuing commands to the GBT M&C system within the time constraints determined by the science goals. Since the TLS Scanner has fixed times for scanning, we anticipate the performance bottleneck to the the TLS Post-Processing component. As such, it may be deployed on and accelerated with GPU hardware.

### 3.4.6 Related Views

TBD

# 4 Relations Among Views

Each of the views specified in Section 3 provides a different perspective and design handle on a system, and each is valid and useful in its own right. Although the views give different system perspectives, they are not independent. Elements of one view will be related to elements of other views, and we need to reason about these relations. For example, a module in a decomposition view may be manifested as one, part of one, or several components in one of the component-and-connector views, reflecting its runtime alter-ego. In general, mappings between views are many to many. Section 4 describes the relations that exist among the views given in Section 3. As required by ANSI/IEEE 1471-2000, it also describes any known inconsistencies among the views.

## 4.1 General Relations Among Views

**CONTENTS OF THIS SECTION**: This section describes the general relationship among the views chosen to represent the architecture. Also in this section, consistency among those views is discussed and any known inconsistencies are identified.

TBD

## 4.2 View-to-View Relations

**CONTENTS OF THIS SECTION**: For each set of views related to each other, this section shows how the elements in one view are related to elements in another.

TBD

## 5 Referenced Materials

| | |
|---|---|
| Barbacci 2003 | Barbacci, M.; Ellison, R.; Lattanze, A.; Stafford, J.; Weinstock, C.; & Wood, W. *Quality Attribute Workshops (QAWs)*, Third Edition (CMU/SEI-2003-TR-016). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03tr016.html>. |
| Bass 2003 | Bass, Clements, Kazman, *Software Architecture in Practice*, second edition, Addison Wesley Longman, 2003. |
| Clements 2001 | Clements, Kazman, Klein, *Evaluating Software Architectures: Methods and Case Studies*, Addison Wesley Longman, 2001. |
| Clements 2002 | Clements, Bachmann, Bass, Garlan, Ivers, Little, Nord, Stafford, *Documenting Software Architectures: Views and Beyond*, Addison Wesley Longman, 2002. |
| ConOps | Bloss et al: *Enhancing GBT Metrology to Support High Resolution 3mm Molecular Imaging for the U.S. Community Concept of Operations*, <https://sharepoint.nrao.edu/pmd/projects/613%20GBT%20Metrology/Systems%20Engineering/Concept%20of%20Operations/613%20GBT%20Metrology%20Concept%20Documents%20v001_3%20181009.docx?Web=1> |

# 6 Directory

## 6.1 Index

## 6.2 Glossary

The following specialized terms are used in the LASSI project.

| Term | Definition |
|---|---|
| GBT observation | The process of obtaining astronomical information using the GBT, or the data generated from that process. (To avoid confusion, please use this in place of "scan", which is ambiguous.) |
| software architecture | The structure or structures of that system, which comprise software elements, the externally visible properties of those elements, and the relationships among them [Bass 2003]. "Externally visible" properties refer to those assumptions other elements can make of an element, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on. |
| TLS | Terrestrial Laser Scanner i.e. the Leica device used in the LASSI project. |
| TLS measurement | A data set about the shape of the GBT's primary or secondary reflector, generated by the Leica laser scanner. (To avoid confusion, please use this in place of "scan", which is ambiguous.) |
| view | A representation of a whole system from the |

| | |
|---|---|
| | perspective of a related set of concerns [IEEE 1471]. A representation of a particular type of software architectural elements that occur in a system, their properties, and the relations among them. A view conforms to a defining viewpoint. |

## 6.3 Acronym List

| | |
|---|---|
| API | Application Programming Interface; Application Program Interface; Application Programmer Interface |
| DAQ | Data Acquisition |
| LASSI | Laser Antenna Surface Scanning Instrument |
| M&C | Monitor and Control |
| SAD | Software Architecture Document |
| UML | Unified Modeling Language |

# Appendix A Frequently Asked Questions

### Q: Why are we calculating Zernike polynomials to model the antenna surface? Wouldn't it be easier to use the data from the scanner to determine the position of the panels, and adjust accordingly?

The ultimate goal of LASSI is to measure the actual shape of the GBT surface and command the M&C system to return it to the desired shape. One solution would be to measure the location of each panel corner, calculate the difference from the desired location, and move the panel by that amount. Unfortunately, this approach relies on extremely precise measurements of individual points on the surface, something that is difficult to achieve in practice.

Instead of attempting to measure individual points, LASSI scans the entire surface, generating a 'point cloud' of several million measurements all over the surface of the GBT. (Note that due to the design and placement of the scanner, these points are not regularly distributed.) While an individual point may not give us the desired precision, the average over a large number of these data points does yield an accurate description of the surface. Using the moving average of the LASSI point cloud to calculate the Zernike polynomial coefficients essentially acts a as an additional low-pass filter, removing noise from the data set and providing a precise calculation of the dish shape.

Additionally, the GBT M&C system can accept Zernike coefficients as inputs when commanded to change the shape of the primary surface, making this step of the LASSI measurement/correction cycle simpler.

### Q: Why do we want LASSI to be able to scan the GBT's subreflector?

Scanning the subreflector and detecting any deformities or flaws allows us to model the entire optical system of the GBT, and use that information to minimize errors in the final signal captured by the receiver. This the GBT primary reflector has an active surface, we can use it to correct for errors in the subreflector. Similar to someone wearing glasses, the first lens (or in our case, prime reflector) corrects any errors introduced by the second lens (the subreflector), so that the distortion is not seen in the final product.

### Q: What is the "point cloud" that Leica refers to? Where is this data stored?

Leica documentation, engineers, and other resources may refer to the "point cloud", which is the data generated by the scanner -- the points it measures. These are stored in the .ptx files generated by the instrument. The .ptx files are flat (plain text) files.

### Q: What is the "noise budget", and what impacts this value?

The terms "noise budget" and "error budget" both refer to the active surface RMS (σ). This value is the amount of deviation in the active surface -- how far it may be from an ideal parabolic shape, beyond which the GBT cannot perform desired high-frequency observations. Many values impact this final number, including (but not limited to) precision and accuracy of the actuators, gravitational and thermal corrections, and the LASSI scans themselves. Currently, the project aims to keep this error budget within 200 microns.